UNIVERSIDADE FEDERAL DO MARANHÃO
ESTRUTURAS DE DADOS I – DEPARTAMENTO DE INFORMÁTICA
LISTA DE EXERCÍCIOS – Revisão de Algoritmos de Matrizes e Vetores

1. Faça um algoritmo que recebe um número inteiro n, e um vetor de inteiros com n posições e retorna a soma dos números lidos percorrendo o vetor.
   Protótipo: int SomaElmVetor(int *v, int n)

2. Faça um algoritmo que recebe um vetor de inteiros e o número n, representando o tamanho do vetor, e retorna:
   a) O maior elemento do vetor.
      Protótipo: int MaiorElmVetor(int *v, int n)

   b) A soma dos números no vetor;
      Protótipo: int SomaElmsVetor(int *v, int n)

   c) O elemento na posição N/2.
      Protótipo: int ElmDoMeioVetor(int *v, int n)

3. Faça um algoritmo que recebe um número n e um vetor v de n posições, e troca os elementos de índice p e q do vetor.
   Protótipo: int TrocaElmsVetor(int *v, int n, int p, int q)

4. Faça um algoritmo para encontrar o número que falta em um vetor de 100 posições que deveria ter os números inteiros de 1 a 100, mas um dos número foi substituído por Zero.
   Protótipo: int NumQueFalta(int *v, int n, int p, int q)

5. Faca um algoritmo que recebe um vetor de números inteiros e retorna o comprimento da sequência consecutiva mais longa existente no vetor.
   Protótipo: int MaiorSequenciaConsecutiva(int *v, int n, int p, int q)

6. Fazer um algoritmo que recebe um número N, e um vetor de inteiros com n posições do vetor, e retorna:

a) o maior valor do vetor;
Protótipo: int ImprimeMaiorValor (int n, int *v);

b) a soma dos números;
Protótipo: int ImprimeSomaDosNumeros (int n, int *v);

c) o elemento da posição teto (n/2)
Protótipo: int ImprimeElementoDoMeio (int n, int *v);

7. Fazer um algoritmo que recebe dois números inteiros N e M, e dois vetores A e B com com n e m posições respectivamente, preenche o vetor C (tamanho igual ao maior entre N e M):
   a) C é o resultado da união do vetor A com o vetor B;

Protótipo: int UniaoVetores(int *a, int *b, int *c, int n)

b) C é o resultado da interseção do vetor A com o vetor B
Protótipo: int InterseçãoVetores(int *a, int *b, int *c, int n)

c) C é o resultado da intercalação do vetor A com o vetor B;
Protótipo: int IntercalaçãoVetores(int *a, int *b, int *c, int n)

d) C contém os elementos 1, 3, 5.. de A;
Protótipo: int ElemEmPosicaoImparNoVetor(int *a, int *b, int *c, int n)

e) C contém os elementos 0, 2, 4.. de B.
Protótipo: int ElemEmPosicaoParNoVetor (int *a, int *b, int *c, int n)

8. Faça um algoritmo que recebe dois números inteiros n e m, e uma matriz de float com n linhas e m colunas m, e retorna a soma dos números da diagonal principal (elementos aijnos quias i é igual a j)
Protótipo: int SomaElmVetor(int *v, int n, int m)

9. Faça um algoritmo que recebe um vetor de n números reais e uma matriz com m linhas e p colunas, de números reais armazenada no vetor vmat, e imprime em quais posições da matriz (linha e coluna) o valor é igual a um dos valores do vetor (imprime também a posição do vetor).
Protótipo: int CmpVectorMatrix (int n, int m, int p, float *vet, float *vmat)

10. Faça um algoritmo que recebe os inteiros n, m, p e q, e três matrizes ma, mb e mc. O algoritmo preenche na matriz mc o produto entre as duas matrizes ma e mb se ele for possível, retornando TRUE e retornando falso caso contrário.
Protótipo: int MultMatrixArmazenadaVetor (int n, int m, int p, int q, float *vma, float *vmb, float *vmc)

11. Faça algoritmo que recebe uma matriz ANxMxP(volume), os valores N, M e P e preenche no vetor vet:

a) Os elementos da linha I da fatia K.
Protótipo: int *DevolveVetorLinhaIFatiaK (int *VolA, int *vet, int N, int M, int P, int I, int K);

b) Os elementos da coluna J da fatia K.
Protótipo: int *DevolverVetorColunaJFatiaK (int *VolA, int *vet, int N, int M, int P, int j, int K);

c) Os elementos da diagonal principal da fatia K.
Protótipo: int *DevolverVetorDiagonalfatiaK (int *VolA, int *vet, int N, int M, int P, int K);

d) Os elementos da linha I de todas as fatias.

Protótipo: int *DevolverVetorLinhaITodasFatias (int *VolA, int *vet, int N, int M, int P, int I, int K);

e) Os elementos (I, J) de todas as fatias.
Protótipo: int *DevolverVetorElmIJTodasasFatias (int *VolA, int *vet, int N, int M, int P, int I, int K);

12. Faça um algoritmo para receber a matriz Anxm(vma), os inteiros n e m, e retornar na matriz vmb a transposta da Matriz A.
    Protótipo: int *transposta(int **vma, int **vmb, int n, int m)

13. Faça um algoritmo que recebe os inteiros n, m, p e q, e três matrizes ma, mb e mc, e retorna em mc o produto entre as duas matrizes.
Protótipo: int *MultMatrixArmazenadaVetor (int n, int m, int p, int q, float *vma, float *vmb, *vmc)

14. Faça algoritmo que recebe uma matriz ANxMxP armazenada num vetor VA, e os valores N, M e P e devolvem um vetor com:
a) Os elementos da linha I da fatia K.
Protótipo: int *DevolverVetorLinhaIFatiaK (int *UA, int N, int M, int P, int I, int K);

b) Os elementos da coluna J e da fatia K.
Protótipo: int *DevolverVetorColunaJFatiaK (int *UA, int N, int M, int P, int j, int K);

c) Os elementos da diagonal principal da fatia K.
Protótipo: int *DevolverVetorDiagonalfatiaK (int *UA, int N, int M, int P, int K);

d) Os elementos da linha I de todas as fatias.
Protótipo: int *DevolverVetorLinhaITodasFatias (int *UA, int N, int M, int P, int I, int K);

e) Os elementos (I, J) de todas as fatias.
Protótipo: int *DevolverVetorElmIJTodasasFatias (int *UA, int N, int M, int P, int I, int K);

15. Faça um algoritmo para receber uma matriz e retornar:
0 – Se for um Matriz Quadrada;
1 – Se for uma Matriz quadrada simétrica;
2 – Se for uma Matriz quadrada Matriz Diagonal;
3 – Se é uma Matriz quadrada Assimétrica;
4 - Se é uma Matriz quadrada Simétrica;

Protótipo: int TipodeMatriz(int **ma, int n, int m)

16. Faça um algoritmo para receber uma Matriz e retornar:
-1 – Se não for uma Matriz quadrada;
0 – Se for um Matriz Quadrada;
Protótipo: int TipodeMatrizQuadrada(int **ma, int n, int m)

17. Faça um algoritmo para receber uma Matriz e retornar:
-1– Se não for uma Matriz quadrada;
0 – Se for um Matriz Quadrada;
1 – Se for uma Matriz quadrada simétrica;
Protótipo: int TipodeMatrizSimétrica(int **ma, int n, int m)

18. Faça um algoritmo para receber uma Matriz e retornar:
    -1– Se não for uma Matriz quadrada;
    0 – Se for um Matriz Quadrada;
    1 – Se for uma Matriz quadrada Matriz Diagonal;
    Protótipo: int TipodeMatrizDiagonal(int **ma, int n, int m)

19. Faça um algoritmo para receber uma Matriz e retornar:
    -1– Se não for uma Matriz quadrada;
    0 – Se for um Matriz Quadrada;
    1 – Se é uma Matriz quadrada Assimétrica;
    Protótipo: int TipodeMatrizAssimétrica(int **ma, int n, int m)

20. Faça um algoritmo para receber uma Matriz alocada como um vetor de vetores e retornar:
    -1– Se não for uma Matriz quadrada;
    0 – Se for um Matriz Quadrada;
    1 – Se é uma Matriz quadrada Simétrica;
    Protótipo: int TipodeMatrizSimétrica(int **ma, int n, int m)

21. Faça um algoritmo para receber uma Matriz e retornar a matriz vmc com o resultado da multiplicação de A por At (sua transposta) ou NULL caso a multiplicação não seja possível:
    a-usando memória adicional (sem copiar a matriz A para uma outra matriz com sua versão transposta);
    Protótipo: int *multiplicaPelaTranspostaA(int **ma, int **vmc, int n, int m)

    b-sem usar memória adicional (copiando a matriz A para uma outra matriz com sua versão transposta);;
    Protótipo: int *multiplicaPelaTranspostaB(int **ma, int **vmc, int n, int m)

22. Faça um algoritmo para receber uma Matriz e retornar o vetor vet com os elementos da linha l da matriz A
    Protótipo: int * linhadaMatriz(int **ma, int *vet, int n, int m, int l)

23. Faça um algoritmo para receber uma Matriz e retornar o vetor vet com os elementos da coluna “p” da matriz
    Protótipo: int * colunadaMatriz(int **ma, int *vet, int n, int m, int p)

24. Faça um algoritmo para receber uma Matriz e retornar o vetor vet com os elementos da diagonal principal da matriz
    Protótipo: int * diagonaldaMatriz(int **ma, int *vet, int n, int m)